

(33)



Kód studenta 39

9 Matematická lingvistika: základy teorie informace (otázka studijního zaměření – 3 body)

Házíme třemi pravými mincemi. Pro každou z nich platí, že pravděpodobnost, že padne panna, je $1/2$. Výsledek hodu reprezentujeme jako hodnotu náhodné veličiny $\langle x_1, x_2, x_3 \rangle$, kde $x_i \in \{P, O\}$.

1. Vypočítejte entropii rozdělení této náhodné veličiny.
2. Jak se entropie změní, jestliže právě jedna ze tří mincí bude falešná a padne na ní vždy panna?

1. možnosti JE 8:

- $P P P$
- $P P O$
- $P O P$
- $O P P$
- $O O P$
- $O P O$
- $P O O$
- $O O O$

KAŽDÁ možnost NASTANE S ROVNAKOU PRAVDĚPODOBNOSTÍ

$$H\left(\frac{1}{8}, \frac{1}{8}, \frac{1}{8}, \frac{1}{8}, \frac{1}{8}, \frac{1}{8}, \frac{1}{8}, \frac{1}{8}\right) = \sum_{i=1}^8 P_i \cdot \log \frac{1}{P_i} = 8 \cdot \frac{1}{8} \cdot \log 8 =$$

= 3

2. $\begin{matrix} P P P \\ P O P \\ P P O \\ P O O \end{matrix} \rightarrow$ PONY 20 SYNTETICKÝ UŽIBA 4 MOŽNOSTI, KTERÉ S PRAVDĚPODOBNOŠTÍ $\frac{1}{4}$. OSYADNÉ MOŽNOSTI MUSE

$$H\left(\frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4}\right) = 4 \cdot \frac{1}{4} \cdot \log \frac{1}{\frac{1}{4}} = 2$$

3b



Kód studenta 39

8 Matematická lingvistika: základní formalismy pro popis přirozených jazyků (otázka studijního zaměření – 3 body)

1. Popište základní fakta o tvorbě jazykových korpusů.
2. Rozdělte jazykové korpusy podle typu značkování, pro každý typ uveďte alespoň dva příklady.
3. Vysvětlete pojem valence.

✓ 1. Jazykový korpus reprezentuje jazyk. Vymíkat z reálných textů (kouzlené články, obsoleta články, krásná čtenářka) obsahuje mnoho (miliónů) slov by reprezentovat celý jazyk, když je pouze jedna výskyt. Korpus je typicky nemenit - abyste pokrajnictví do výskytu.

2. Môžu byť značkované

a) morfológickej

Teda slováky vo výzvach je priradené morfológickej značke, ktorá popisuje jeho gramatické kategórie. Značky sú priradené v kontexte textu.

b) syntaktickej PRÍKLADY: Pražský závislostný korpus, Český národný korpus,

v korpusoch sú výzvy reprezentované s závislosťami, resp. složkovými strukturami, pozdĺžne teda vzájomne spojené.

Môžeme teda skúmať vzájomnosti medzi slovmi vo výzvach.

PRÍKLODY - PBN TREEBANK

Pražský závislostný korpus

3. ~~VET~~

VALÈNCIA JE "SCHOPNOSÍ" SCOV NA SE BA VÍAZKÝ INÉ
SCOVÁ.

NAPRÍKLAD: PRŠÍ.

\ NEJMÍSTI VÍZKÁJÍ ZIAONE SCOVO

KAPROVY ROKU DAROVÁŘ VÍZKÉ HZ TKI SCOVÁ
JE DÝTRANZITNÉ

PETER DAROVÁL MARCELE KVET

KYO DAROVÁL ?
ČO DAROVÁL ?
KONC TO DAROVÁL ?,

THEORIA LÉČENIE ROZLIŠUJE AICHTANTOV - VÝVĒRÍUS VÄZBY
A VOLNÉ DOPLŇNIA - ~~NAPRÍKLAD~~
OKOLNOSTI

25

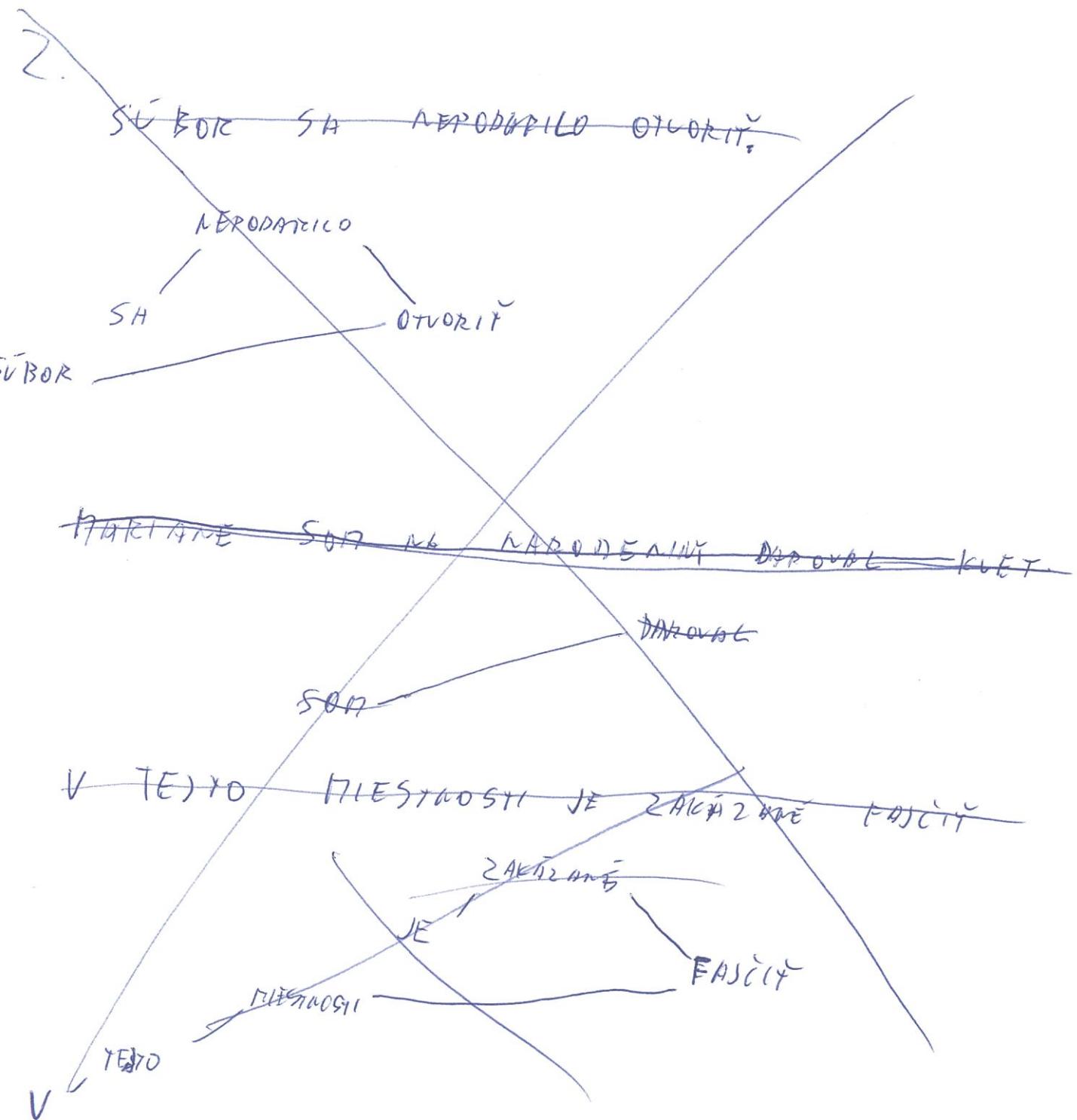


Kód studenta 39



- 7 Matematická lingvistika: formální jazyky a automaty (otázka studijního zaměření – 3 body)

1. Vysvětlete pojem neprojektivity v závislostních stromech.
 2. Uveďte příklady dvou neprojektivních vět.



reprezentácia

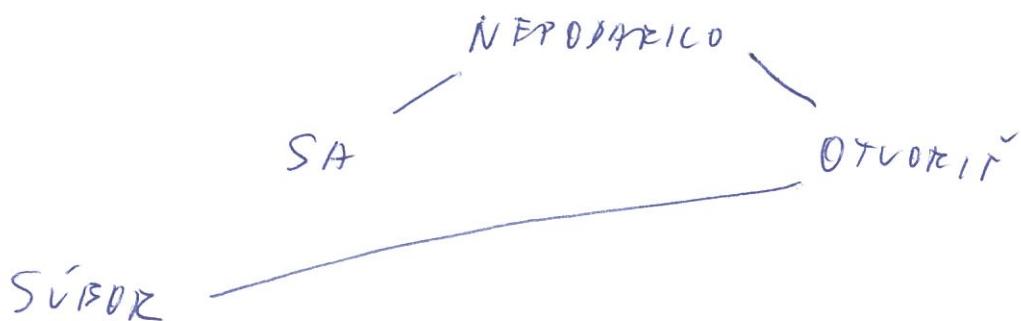
NEPROJEKTÍVNY ZÁVISLOSTNÝ STRON KEDIEĽA VRECHOLOU
STRONU "PRESKOČÍ" ČASŤ PODSTRONU OYCA VRECHOLOU.
(FORMA)

V STRONE SA TO PREDALI AŽUNOV PREKRI ŽEHOV (POPOD)
SLOVĀ.

• TAKO TO NEPROJEKTÍVNU KONŠTRUKCIU NEDOKÁŽEME "VZÁŤVORKO"
~~NE~~ A TEDA PREVIESŤ NA ZLOŽKOVÝ STRON.

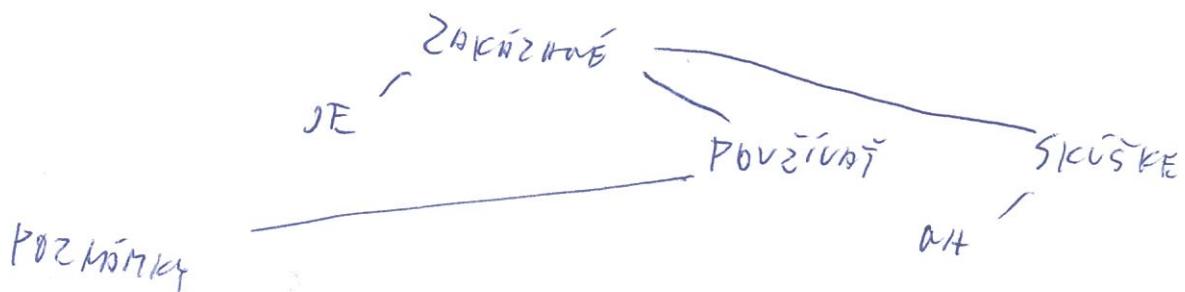
2.

SÚBOR SA NEPODDALO OTVORET



~~DVAKA~~ ~~SA~~ ~~NEPODAMICO~~ ~~OTVORET~~

Poznávatec JE ZAKLÍCHANÉ POUŽÍVATEĽ MA SKUŠKE.





Kód studenta 39



6 Optimalizační metody (3 body)

Nechť $Ax \leq b$ je systém lineárních nerovnic o n proměnných. Vynásobením každé nerovnosti kladnou konstantou můžeme docílit, že první sloupec matice A je vektor obsahující pouze složky 0, -1 and 1. Systém $Ax \leq b$ můžeme tudíž ekvivalentně zapsat jako

$$\begin{aligned} a'_i x' &\leq b_i \quad (i = 1, \dots, m_1), \\ -x_1 + a'_j x' &\leq b_j \quad (j = m_1 + 1, \dots, m_2), \\ x_1 + a'_k x' &\leq b_k \quad (k = m_2 + 1, \dots, m), \end{aligned}$$

kde $x' = (x_2, \dots, x_n)$ a kde a'_1, \dots, a'_n jsou řádky A bez první složky. Pak se můžeme zbavit x_1 : dokažte, že systém $Ax \leq b$ má řešení právě když systém

$$\begin{aligned} a'_i x' &\leq b_i \quad (i = 1, \dots, m_1), \\ a'_j x' - b_j &\leq b_k - a'_k x' \quad (j = m_1 + 1, \dots, m_2, k = m_2 + 1, \dots, m) \end{aligned}$$

má řešení. Ukažte, že opakováním použitím tohoto kroku je možné vyřešit systém lineárních nerovnic (nebo dokázat, že systém nemá řešení). Jaká bude časová složitost takového postupu?

$\forall j = m_1 + 1, \dots, m_2$ můžeme užití výpočtu $-b_j - a'_j x' \leq x_1$, TEDA

$$a'_j x' - b_j \leq x_1$$

$\forall k = m_2 + 1, \dots, m$ můžeme užití $-a'_k x' \leq x_1$, TEDA

$$x_1 \leq b_k - a'_k x'$$

2. YONO VÝPLÝVÁ že pro všechny dvorce $j = m_1 + 1, \dots, m_2$
 $k = m_2 + 1, \dots, m$

$$\text{Plati } a'_j x' - b_j \leq x_1 \leq b_k - a'_k x'$$

to můžeme dále upravit: ✓

$$a_1' x^1 + a_2' x^1 \leq b_1 + b_2$$

$$(a_1' + a_2') x^1 \leq b_1 + b_2$$

YEN DOSIAČTIE NOVÝ SYSTÉM $A'x \leq c$, ~~AKO~~

AKO A' MA O JEDEN STUPEC MENEJ AKO A , TO MA
 $\sigma (m_1 - m_2) \cdot (m_2 - m_3) - (m_1 - m_3)$ RIADKOV VIAC.

✓

V ĎALŠOM KROKU VŠAK ZLOUČ MOŽNÉ A VBRAT JEDEN
STUPEC A PRIAŤ RIADKY, AŽ DOSTANEM TRIUÁČKE RESÍTIA
PRE NEROVNICE O 1 NEZMINU. TÚTO VYRIEŠIŤ A
BUDENE SPÄŤME DOŠAŤOVAT.

V KROKOM KROKU SA ČRTE ZVÝSOTIÝ POČET RÔDNIC,
INSONÁ ZLOŽITOSŤ BUDÉ $O(m^n) = n$ KRÍTY UDETÍNE
STUPEC, ~~VŽDYM~~ DOSIA

DAĽKE MÔŽE AŽ m KRÍTÝ ZVÄZIŤ,
V KROKOM KROKU SA VELIKOSŤ
co to je m
V i-TEM KROKU?

* POTREBA ZDÍMONIŤ, ŽE PREDMET' SOUTAVA MA'
RESÍM! (\Leftrightarrow NOVA' MA' RESÍM!

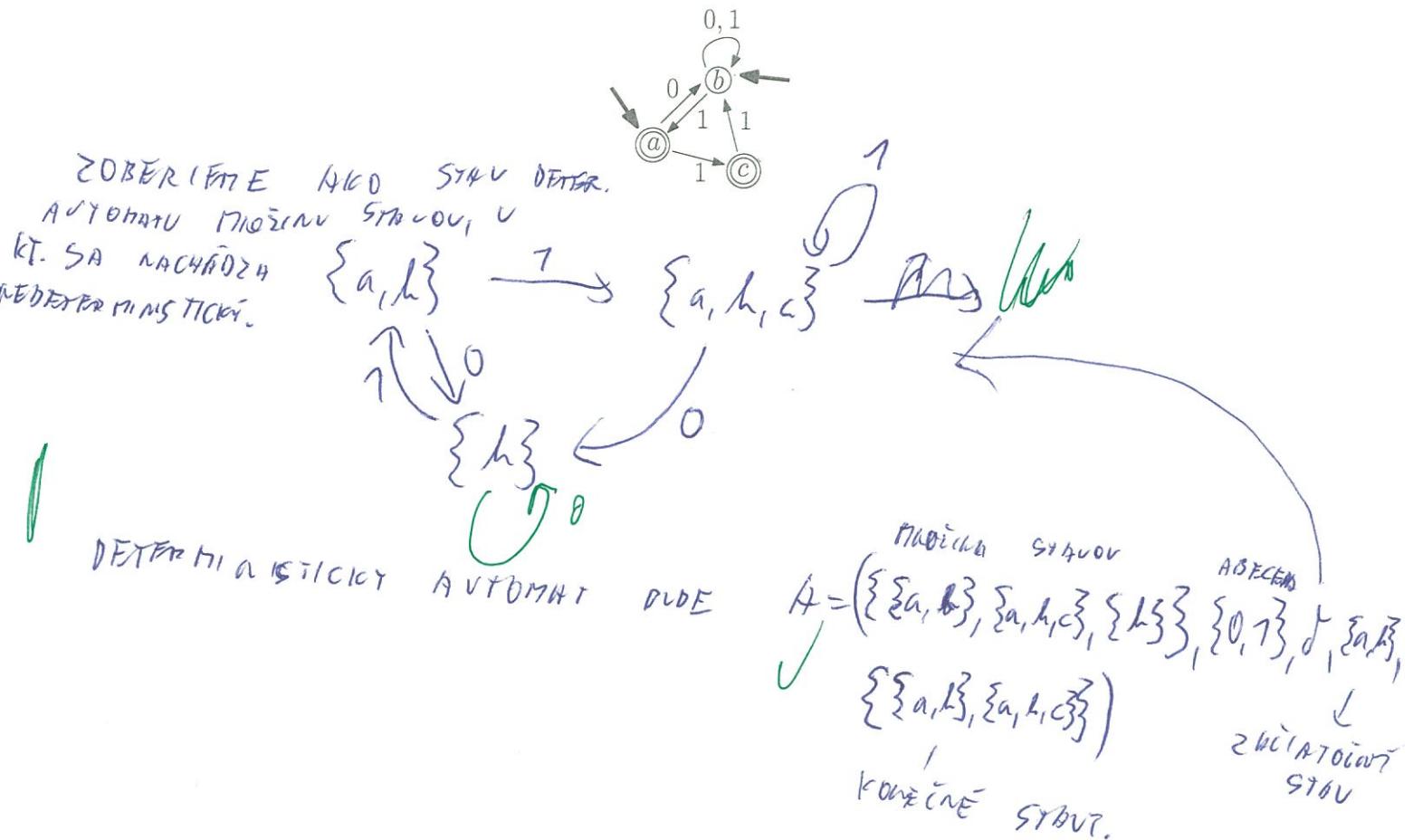
3



Kód studenta 39

5 Automaty (3 body)

- Převeďte nedeterministický konečný automat $A = (\{a, b, c\}, \{0, 1\}, \delta, \{a, b\}, \{a, c\})$ z obrázku na ekvivalentní deterministický automat.
- Do jakých tříd jazyků v Chomského hierarchii patří jazyk $L(A)$?



2. ~~HIER~~

DETERRIMENSYKCE A NEDETERMINISTICKÉ AUTOMATY ~~NE~~
PRIJIMAJÚ ROVNAKÝ MOŽNÝ JAZYKOV, ~~VIENE PRE~~

JE PRIJÍMAMY KONEČNÝ AUTOMAT. ... a dôlnich

(3 -) 02



Kód studenta 39

3 Objektový interface pro REST API server (3 body)

Mějme specializovaný web server pro REST API implementovaný v mainstreamovém objektově orientovaném staticky typovaném jazyce (C++, C# nebo Java) s následujícími vlastnostmi. Server umožňuje, aby se do něho dynamicky vkládaly služby, přičemž služba je objekt třídy implementující rozhraní `IService`, který zapouzdřuje několik REST metod. Služby jsou identifikovány řetězci, které musí být unikátní v rámci serveru, REST metody jsou rovněž identifikovány řetězci, které musí být unikátní v rámci dané služby. Součástí rozhraní třídy `IService` je i schopnost publikovat seznam svých REST metod. Souvislost mezi REST metodami a HTTP metodou volání (GET, POST) neřešíme, v dalším kontextu je metoda bez přívlastku chápána jako členská funkce třídy.

Zpracování požadavku přes REST API probíhá tak, že server přeče URL a další data z HTTP protokolu a z nich dekóduje identifikátor služby, identifikátor REST metody a kolekci parametrů. Dle identifikátorů najde ve vnitřních záznamech požadovanou službu a na ní zavolá metodu implementující požadovanou REST metodu, přičemž metoda dostane kolekci parametrů jako vstupní argument a vrátí řetězec. Pro naše účely si představme kolekci parametrů jako mapu/slovník (konkrétní typ si upravte dle zvoleného jazyka), kde klíče jsou řetězce (názvy parametrů) a hodnoty jsou objekty typu `ParameterValue` (další detaily jsou pro naši úlohu nezajímavé).

Server je reprezentován objektem třídy `Server`. Tato třída má (mimo jiné) dvě podstatné metody, které by mohly být symbolicky zapsány přibližně takto (přesný zápis závisí na použitém jazyce):

```
public void add(IService service)  
private string dispatch(string serviceId, string methodId, ParametersCollection parameters)
```

Metoda `add` zaregistrouje novou službu v rámci serveru. Tato metoda může být relativně pomalá a volá se typicky jen při startu aplikace (např. když jsou načítány základní moduly). Metoda `dispatch` najde a zavolá cílovou metodu příslušné služby a je volána výhradně z vnitřní implementace serveru při zpracování požadavku ze sítě. Metoda `dispatch` by neměla mít velký overhead.

1. Navrhněte, jak by měl vypadat interface `IService` a stručně (např. komentářem) vysvětlete význam jednotlivých metod (pokud není naprosto zřejmý z názvu metody).
2. Představte příklad konkrétní třídy implementující rozhraní `IService`. Zaměřte se zejména na realizaci části rozhraní zodpovědné za předání informací o nabízených metodách.
3. Stručně (případně v pseudokódu) popište, jak bude vypadat implementace metod `add` a `dispatch` třídy `Server`. Pokud tyto metody potřebují přistupovat ke členským proměnným třídy, popište tyto proměnné také. Připomeňme, že metoda `dispatch` by měla být rozumně efektivní.

Drobné chyby v syntaxi budou tolerovány, avšak celkový návrh a logika rozhraní by měly obecně odpovídat principům a zvyklostem zvoleného jazyka. V jazycích, které to umožňují, je povoleno rozumné použití reflexe, nicméně reflexe není nutně vyžadována.

V ULOŽÍČKACH KÓDU 16.10.2015 RODIEČ

INTERFACE IService

{

STRING ID { GET; }

IMPLIMENTABLE<(METHOD)> LIST METHODS();

METHOD GET METHOD(STRING METHOD ID);

REST // VZÁTÍ METODU S DBAŘÍM ID.

INTERFACE METHOD // REPREZENTUJE JEJÍKU REST METÓDU.
 STRING ID {GET;} → Objektivně složité - každá metoda
 musí mít odvozenou třídu.
 void INVOKE(PARAMETERS COLLECTION PARAMETERS);
 ENUMERABLE<TYPE> HEAD PARAM TYPES {GET;} ↑
 ↑
 ↑ REPREZENTUJE (GPOLO S ID)
 ALAVICKU METÓDY, SÚ TAK
 TYPY PARAMETROV ↑
 ↑ SPUSTÍ REPREZENTACI
 METÓDU S DANÝMI
 PARAMETRAMI.

2.

CLASS SERVICEIMPL

```

{ PUBLIC SERVICEIMPL(STRING ID)
{
  THIS.ID = ID;
  LIST METHODS(); - METODNÍ METÓDY
}
  
```

DICTONARY<STRING, METHOD> METODY = new DICTIONARY<STRING, METHOD>();

ENUMERABLE<METHOD> LIST METHODS()

```

{ RETURN METODY. VALUES;
}
  
```

VRÁTI VŠETKY HODNOTY
ZO SLOVNÍKA METÓDY.

METHOD GETMETHOD(STRING ID)

```

{ :
  RETURN METODY[ID];
}
  
```

'co když chybí?'

3.

V TŘINĚ SERVER BUDE SLOVNÍK DICTIONARY<STRING, ISERVICE>
 METÓDA APP BUDE DO NEBO JEDNOU DO PRIDÁVAT
 SERVICES

S TÝM, JEŽ MAŁA SKONTROLOVAT UNIKATNOSŤ SERVICE ID

KÓD 39
ÚLOHA 3

```
VOID ADD (STRING ID)
{
    VOID ADD (SERVICE SERVICE)
    {
        IF (SERVICES.KEYS CONTAINS (SERVICE.ID))
            THROW NOTUNIQUEEXCEPTION();
        ELSE
            SERVICES.ADD (SERVICE.ID, SERVICE);
    }
}
```

Nejprve musí mít service!

DISPATCH máže metodu a zavolá je invoke, je potřeba
OŠTŘITI, že metoda s držím ID, ani service neexistuje
: VISITATE, že existuje service ID a) methodID v ránci
SERVICES[SERVICE.ID].[✓] GETMETHOD (METHOD ID). INVOKE (PARAMETERS),
:

1 SERVICE

4 Kil. 470⁹

{
 ~~(EXTRACTABLE(METHOD))~~ LIST METHODS ();

$8 \cdot 10^{12}$

String ID {get; }
METHOD GET METHOD (IF STRING METHOD ID)

4.4h

4 MB.

METHOD

65 536

{
 String ID
 void INVOKER(Parameter COLLECTION)

$2^{10} B = 2^{16}$

2^{26}

7 GND JE

3 body



Kód studenta 39

4 Souborový systém (3 body)

Tato otázka obsahuje zjednodušený popis souborového systému FAT. Pokud chcete, můžete v odpovědi uvažovat i skutečný souborový systém FAT (pokud ano, výslově to napište).

Oddíl naformátovaný (zjednodušeným) souborovým systémem FAT12/16/32 je rozdělený na 3 části:

1. boot sektor (nultý logický sektor oddílu),
2. tzv. tabulka FAT, viz dále (1. až N. sektor oddílu),
3. datové sektory (N+1. sektor až poslední sektor oddílu), které obsahují samotná data souborů (nebo adresářů, nicméně jelikož ty se z pohledu alokace sektorů od souborů v souborovém systému neliší, tak se jimi dále nebude zabývat zvláště). Pro jednoduchost předpokládejme, že alokační jednotka souborového systému je právě jeden sektor disku. Sektoru disku mají jednu z obvyklých velikostí – označme ji jako X bytů.

Každý datový sektor oddílu je přiřazený právě jednomu souboru, nebo je označený jako volný. Každý soubor na disku zabírá určité množství celých sektorů, přičemž ale tyto sektory nemusí být konkrétnímu souboru přiděleny kontinuálně (soubory mohou být na disku fragmentované). V adresářovém záznamu pro konkrétní soubor je zapsáno číslo datového sektoru (číselovány od 1), který obsahuje data prvních X bytů souboru (pro soubory s velikostí menší nebo rovnou X bytů je to zároveň poslední sektor souboru). Pro soubory s velikostí větší než X bytů nalezneme informaci o dalších sektorech přidělených souboru ve FAT tabulce oddílu – pro každý soubor (který zabírá např. M sektorů) obsahuje FAT tabulka „zakódovanou“ obdobu jednosměrně vázaného seznamu posloupnosti čísel 2. až M. sektoru souboru. Přesný obsah FAT tabulky je následující:

Pro každý datový sektor obsahuje FAT tabulka právě jeden Z bitový záznam ($Z = 12$ bitů pro FAT12, 16 bitů pro FAT16, 32 bitů pro FAT32) – tento záznam je bezznaménkové Z bitové celé číslo uložené v pořadí little endian. V tabulce FAT je tedy právě tolik záznamů, kolik oddíl obsahuje datových sektorů, a žádné jiné informace FAT tabulka neobsahuje (FAT tabulka je tedy fakticky jen pole celých čísel). Záznam na offsetu 0 v prvním sektoru FAT tabulky obsahuje informaci o 1. datovém sektoru oddílu, hned za ním (bez paddingu) následuje záznam pro 2. datový sektor oddílu, pak záznam pro 3. datový sektor oddílu, atd. Pokud záznam pro datový sektor A obsahuje číslo 0, tak je tento datový sektor volný a není přidělen žádnému souboru. Pokud záznam pro datový sektor A obsahuje číslo B, tak je datový sektor A přidělený nějakému souboru, a po X bytech dat toho souboru uložených v datovém sektoru A je následujících X bytů souboru uložených v datovém sektoru B (kde pro fragmentované soubory nemusí být B rovno A+1, a obecně může být B být větší i menší než A). Pokud záznam pro datový sektor A obsahuje maximální hodnotu Z bitového čísla, pak je sektor A posledním sektorem nějakého souboru (M. sektorem přiděleným souboru o velikosti M sektorů). Pokud tedy např. data souboru A.TXT budou ležet v datových sektorech 10, 7, 8, 15, tak v adresářovém záznamu pro A.TXT bude zapsáno číslo 10 a ve FAT tabulce bude v záznamu pro sektor 10 uloženo číslo 7, v záznamu pro sektor 7 číslo 8, v záznamu pro sektor 8 číslo 15, a v záznamu pro sektor 15 maximální hodnota Z bitového čísla.

1. Jaká může být obvyklá hodnota X? Pokud je velikost oddílu právě 1 GiB, je vhodné tento oddíl naformátovat souborovým systémem FAT16 nebo FAT32? Vysvětlete proč.
2. V takovém 1 GiB oddílu máme uloženo 100 000 souborů, kde každý má velikost 1 KiB. Pokud nyní chceme přečíst obsah všech těchto souborů, bylo by pro typický disk vhodné načíst si předem celou FAT tabulku do paměti? Nebo by bylo vhodnější před čtením každého ze souborů znova načíst do paměti pouze ty sektory FAT tabulky, o kterých zjistíme, že obsahují informace potřebné pro právě čtený soubor? Vysvětlete proč.
3. Předpokládejte, že v „globální proměnné“ (resp. statické proměnné nějaké třídy) **fat** typu pole bytů máme načtené veškeré záznamy celé FAT tabulky oddílu naformátovaného souborovým systémem FAT12 – jeden záznam FAT tabulky má velikost 12 bitů, tedy každé 3 byty pole **fat** obsahují právě 2 záznamy o 2 datových sektorech (záznam pro sektor s nižším číslem je vždy v 1. bytu a ve spodních 4 bytech 2. bytu; záznam pro sektor s vyšším číslem je ve vyšších 4 bytech 2. bytu a ve 3. bytu). V jazyce C# nebo Java nebo C++ naprogramujte proceduru, která jako svůj jediný argument dostane číslo prvního datového sektoru přiděleného nějakému souboru, a má na standardní výstup vypsat čísla všech datových sektorů tomuto souboru přiřazených (dle informací v proměnné **fat**).

1. OBLÍKLÁ HODNOTA X JE ✓ nebo 4KB
1 GiB = 2^{30} B 572 B - 8 KiB

$$\text{FAT 16} \quad s \quad x = 2^{16} \cdot 8\text{KiB} = 2^{13} \text{ B} \quad \text{DOKUZÉ ADDRESSOVÝ}$$

$$2^{16} \cdot 8\text{KiB} = 2^{16} \cdot 2^{13}\text{B} = 2^{29} \text{ B} \quad \text{TO JE MÉRÉJ AKO } 2^{26}\text{B}$$

$$\text{FAT 32} \quad \text{PRE} \quad x = 572\text{B} = 2^9 \text{ B}$$

$$2^{32} \cdot 2^9\text{B} = 2^{41} \text{ B} \quad \text{TO JE VÍCE AKO } 2^{36}\text{B}$$

FAT 32 BY SME NOMU POUŽIT ✓

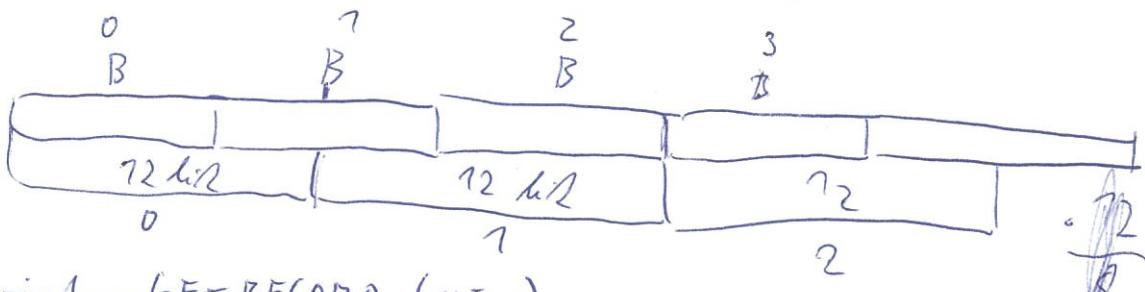
2.

~~KOLO BY VHOДESИE SI NАČÍTAť~~

AK PREPOUKUJÚME, ŽE $x \geq 7\text{KiB}$, POTOM

NEDÁ ZMYSEL NCIÍTAŤ SI TABUĽKU DO RAMÄTI, PRETIAŽ
JU VÔBEC REPOUŽIJEME. BUDÉ NÍS ZAKJÍMAŤ VÝSOKÝ IBH
PRVÝ (JEDINÝ) SEKTOR SÚBORU, A TEN JE ZAPÍSAVÝ V
ADRESÁRI.

AK $x \leq 7\text{KiB}$, POTOM ŠA TO PRAHODEPODOBNE OPLATI,
AK (NECHANICKY) HOD NEMUSET KOMITAT A PRESUVAT HLAČU
NEDZI RÔZNE ČASÍAMI DISKU. ✓



int GET_RECORD (int i)

{

int DIV = i / 3;

int MOD = i % 3;

if (MOD = 0)

return;

int GET RECORD (int i) 1C0D 39
U C014A4
 {
 int result = 0;
 int lbyte = i * 3 / 2;
 int rbyte = i * 3 % 2;
 if (lbyte >= 128)
 {
 result = FAT[lbyte] << 4;
 result += (FAT[lbyte + 1] & 0xF0) >> 4;
 }
 else
 {
 result = (FAT[lbyte] & 0x0F) << 8;
 result += FAT[lbyte + 1];
 }
 return result;
 }

void WRITE FILE RECORDS (int I)

{
~~FILE~~ CONSOLE.WRITELINE(I);
 while (FOR (i))
 {
 I = GET RECORD (I);
 if (I == 0xFFFF)
 RETURN;
 CONSOLE.WRITEREC (I);
 }

26
kop



Kód studenta 39



2 Databáze (3 body)

- Načrtněte, jak byste v databázové tabulce reprezentovali binární vztah M:N. Lze z definice tabulky $T(FK_1, FK_2)$, vzniklé převodem binárního vztahu, určit jeho původní kardinalitu (1:1, 1:N, M:N)? Vysvětlete.
- Uvažujte transakce $T_1: W(A) R(B) W(C)$ a $T_2: R(A) W(B) R(C)$. Je rozvrh $S: W_1(A) R_2(A) R_1(B) W_2(B) W_1(C) R_2(C)$ konfliktově serializovatelný (conflict serializable)? Vysvětlete proč a pokud není, navrhněte úpravu, aby byl.

1. Budeme mít 3 tabulky. 2 z nich reprezentují objekty, ~~A, B~~ atributy nedzi ktorí je binárni vztah zo zadania.

$$\begin{aligned} \text{OBJEKT } A & (ID, Attrib_1, Attrib_2, \dots, Attrib_c) \\ \text{OBJEKT } B & (ID, AttrA_1, AttrA_2, \dots, AttrAd) \end{aligned}$$

Objekt A má c atributov, objekt B má d atributov v tabuľke je kód ID - identifikátor objektu.

Tretia tabuľka bude reprezentovať vzťah reláciu nedzi nimi:

$$R(ID_A, ID_B) \quad \checkmark$$

$$FK: ID_A \subseteq A.ID$$

$$ID_B \subseteq B.ID$$

ID_A, ID_B sú evidenčné kódy.

Z DEFINICIE SA TO NEHÁ VZIŤ.

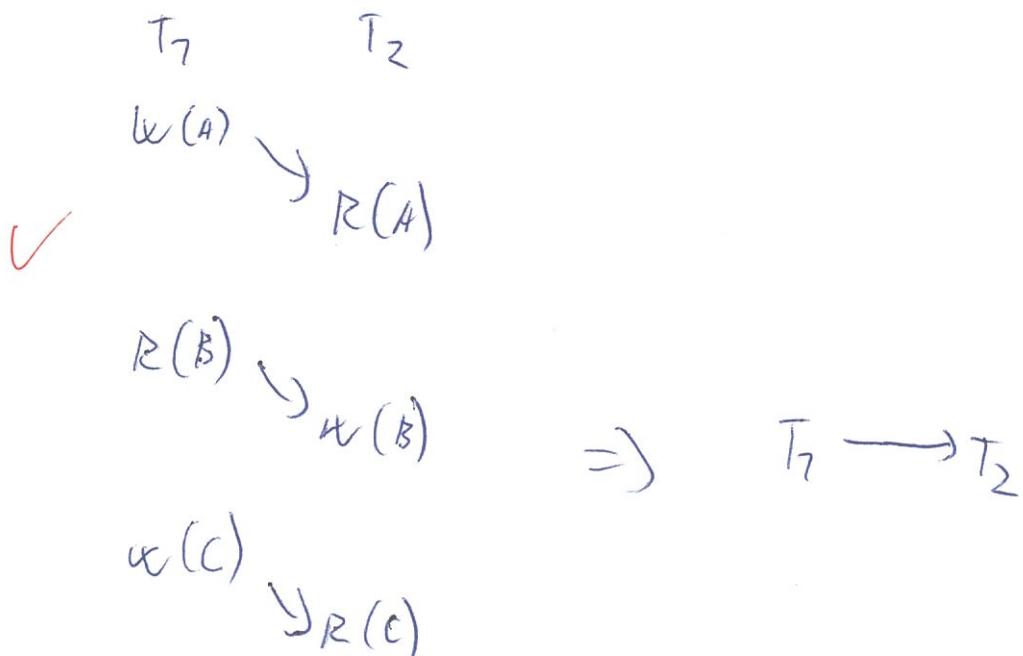
TÓ MÁN IBÁ HODNOTI, ŽE KEDOM DUOJICH HODNOT
 (FK_1, FK_2) JE V TABUĽKE IBÁ IZAZ, TO VŠAK
PLATÍ PIRE VŠETKÝ CHARACTERTY, ale pro M:N
1:1 by mela $(\underline{FK_1}, \underline{FK_2})$

~~Ak B je podmienkou T($\underline{FK_1}, \underline{FK_2}$)~~, ~~TNC je možnosť~~
~~ak VILČÍ M:N~~

1:N by mela $(\underline{FK_1}, \underline{FK_2})$
resp. $(\underline{FK_1}, \underline{FK_2})$

Z.

ROZVRH JE SERIALIZOVATELNÝ



PÔŽEME MASKA PREVIESŤ T_1 A POZOR
 T_2 .

③ MM



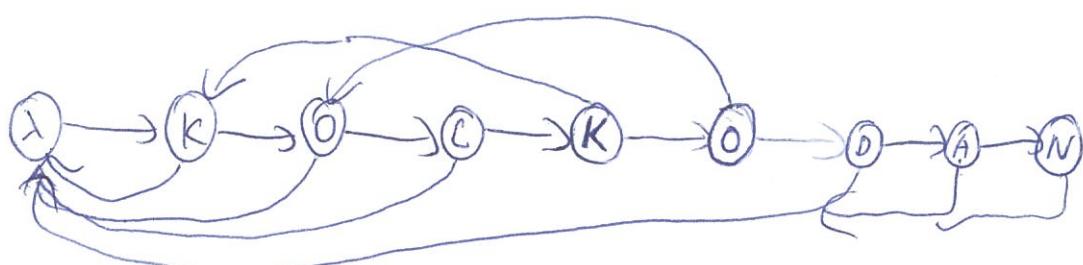
Kód studenta 39

~~new! odnášla i hola 2~~

1 Algoritmy a datové struktury: vyhledávání v textu (3 body)

- Definujte vyhledávací automat algoritmu KMP (Knuth-Morris-Pratt). Jak vypadají jeho stavy, dopředné a zpětné hrany?
- Nakreslete vyhledávací automat pro slovo KOCKODAN.
- Jakou časovou složitost má konstrukce automatu a jakou jeho použití k vyhledání všech výskytů slova v textu?

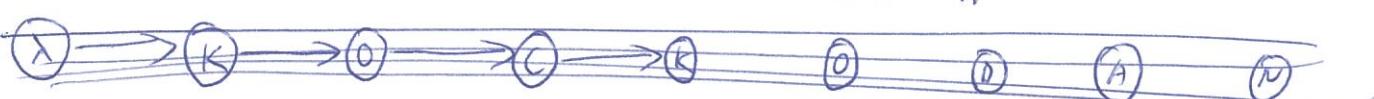
②



✓

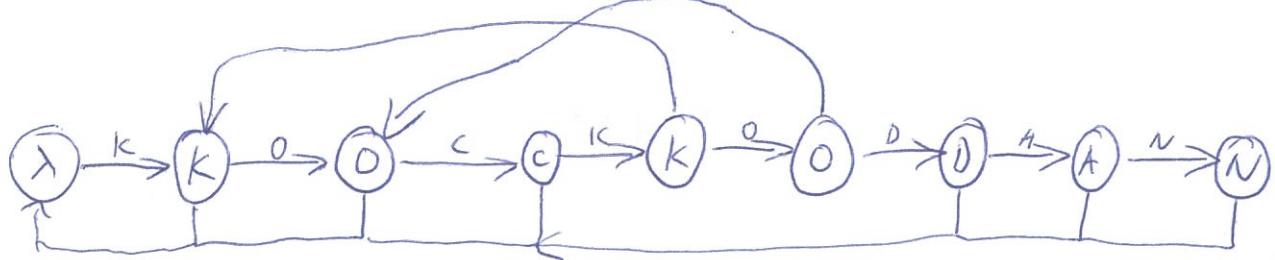
7.

DANÉ ~~TEXT T, A RĚZEC T (TEXT) A RĚZEC S (VYHLEDÁVANÉ SLOVO,~~
 KNP AUTOMAT CHAČAKYERIZUJE SLOVO THK, ABY ~~SHE~~ ~~SH~~ KNA
 PRI VYHLEDÁVÁNÍ V TĚSTE ~~STÁČETE~~ ~~VĚDÝ~~ VĚDÝ VEDEC
 POUZE V KONSTANTNÍ ČASE, AKÉ NADLHOŠÍ. SUFFIX DO
 DOPISAT PREHĽADATELHO TEXTU JE AJ PREFIXOM SLOVA S.
 DÁ JEDEN SÍAV PRE KAŽDE ~~ZNAK S~~ ~~PISMA~~, A SÍAV λ , KTORÝ
 SYMBOLIZUJE, λ SUFFIX DĽŽKY 0. DOPREDEM HRANI VĚDÝ
 MEDZI SUSEDIACIMI ~~PISMA~~ ZNAKMI.



SPÄTNÁ HRANA VĒDIE i DO j , AK PODREZACE
 SPÄTNÁ HRANA VĒDIE i DO j -TEHO ZNAKU DO j -TEHO, TEDA
 2 $S[i] \rightarrow S[j]$, AK S_R ^{ROVNAŠÚ} PODREZACE $S[0,j] = S[i]$
 4 ZHKOVENÍ i JE MAXIMAČNE MOŽNÉ. ✓ $= S[i-j, i]$

2.



3. Abydňat sa dôľžitá poslanka v čase L , nájdeme od dĺžky slov s $O(s)$,

Vyhľadávame ktoré súvahy v čase $O(1)$, keďže je dĺžka textu.

Celkovo $O(s+L)$.

